НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»
Теплоенергетичний факультет
Кафедра автоматизації проектування енергетичних процесів і систем

До захисту допущено:
Завідувач кафедри
_____ Олександр КОВАЛЬ
«___»_____2021 р.

Дипломна робота

на здобуття ступеня бакалавра
спеціальності 122 «Комп'ютерні науки»
освітня програма «Комп'ютерний моніторинг та геометричне моделювання процесів і систем»
на тему: «Автоматизована система з обліку даних абітурієнтів»

Виконала:
студентка IV курсу, групи ТР-72
Рутковська Аліна Олегівна _____

Керівник:
Ст. викладач
Колумбет Вадим Петрович _____

Рецензент:
к.т.н.,
Баранюк Олександр Володимирович _____

Засвідчую, що у цій дипломній роботі немає запозичень з праць інших авторів без відповідних посилань.
Студентка _____

Київ — 2021 року

Національний технічний університет України
“Київський політехнічний інститут імені Ігоря Сікорського”

Факультет теплоенергетичний

Кафедра автоматизації проектування енергетичних процесів і систем

Рівень вищої освіти перший рівень

спеціальності 122 «Комп’ютерні науки»

освітня програма «Комп’ютерний моніторинг та геометричне моделювання процесів і систем»

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Олександр КОВАЛЬ
(підпис)
”___”_____2021р.

ЗАВДАННЯ
на дипломну роботу студенту

Рутковській Аліні Олегівні
(прізвище, ім’я, по батькові)

1. Тема роботи _Автоматизована система з обліку даних абітурієнтів

керівник роботи_____ Колумбет Вадим Петрович, ст. вик.
(прізвище, ім’я, по батькові науковий ступінь, вчене звання)

затверджена наказом вищого навчального закладу від ”24” травня 2021р. № **1267-с**

2. Строк подання студентом роботи _____

3. Вихідні дані до роботи _мова програмування C#, фреймворк ASP.NET Core, середовище розробки Visual Studio

4.Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)

дослідити вимоги членів приймальної комісії, щодо автоматизації роботи, розробити концепцію системи, що відповідатиме вимогам, реалізувати програмний продукт

5. Перелік ілюстративного матеріалу
схеми архітектури програмного продукту, знімки інтерфейсу

6. Дата видачі завдання ”_10_”_____жовтня_____ 2020 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів виконання дипломної роботи	Термін виконання етапів роботи	Примітки
1.	Затвердження теми роботи	9.10.2020	
2.	Вивчення та аналіз задачі	16.03.2021-01.04.2021	
3.	Розробка архітектури та загальної структури системи	02.04.2021-13.04.2021	
4.	Розробка структур окремих підсистем	14.04.2021-25.04.2021	
5.	Програмна реалізація системи	26.04.2021-09.05.2021	
6.	Оформлення пояснювальної записки	10.05.2021-22.05.2021	
7.	Захист програмного продукту	11.05.2021	
8.	Передзахист	25.05.2021	
9.	Захист	14.06.2021	

Студент _____ _____Рутковська А.О._____
(підпис) (прізвище та ініціали,)

Керівник роботи _____ _____Колумбет В.П._____
(підпис) (прізвище та ініціали,)

АНОТАЦІЯ

У роботі розглянуті теоретичні та практичні аспекти розробки програмного продукту для автоматизації системи з обліку даних абітурієнтів.

Метою роботи є створення веб-застосунку який спростить роботу приймальної комісії.

Для досягнення мети систематизовано інформацію про роботу приймальної комісії, розроблено систему, що автоматизує певні завдання приймальної комісії, протестовано веб-застосунок на зручність інтерфейсу.

Пояснювальна записка складається зі переліку умовних позначень, вступу, чотирьох розділів, висновку, списку використаних джерел; містить 33 сторінки, 10 рисунків та 2 додатки. Список використаних джерел включає 12 бібліографічних найменувань за «Переліком посилань».

Ключові слова: веб-застосунок, приймальна комісія, абітурієнт, MVC шаблон, ASP.NET Core.

ABSTRACT

The paper considers theoretical and practical aspects of software development for automation system for accounting data of entrants.

The purpose of this work is to create a web application that will simplify the work of the selection committee.

To achieve the goal, information about the work of the selection committee was systematized, a system that automates certain tasks of the selection committee, was developed, the web application for the convenience of the interface was tested.

The explanatory note consists of an introduction, four chapters, conclusion, list of used sources; contains 33 pages, 10 figures, and 2 applications. The list of used sources includes 12 bibliographic items.

Key words: web application, admissions office, entrant, MVC, ASP.NET Core.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

ПК	—	Приймальна комісія
ОС	—	Операційна система
БД	—	База даних
EF	—	Entity Framework
VS	—	Visual Studio
СУБД	—	Система управління БД
UI	—	User interface (Користувацький інтерфейс)

ВСТУП

Щороку приймальна комісія вимагає багато людських ресурсів. Процес прийняття та систематизації документів абітурієнтів супроводжується використанням кількох окремих баз даних дані у яких частково дублюються. Окрім того працівникам доводиться в ручну співставляти дані з різних джерел для того щоб отримати певні необхідні звіти і статистику.

Для вирішення цієї проблеми може бути запроваджена веб-система з наступним її інтегруванням на факультеті.

Така система обліку даних абітурієнтів є актуальною для будь-якого вищого навчального. Її головними перевагами є централізований доступ до сервісу через веб-браузер, об’єднання усіх даних про абітурієнтів в одному джерелі, зручний спосіб пошуку інформації та можливість автоматичного створення звітів і підбиття статистики.

Данна система веб-застосунку дозволятиме працівникам приймальної комісії створювати особові справи для абітурієнтів у єдиній формі, переглядати загальний список абітурієнтів, здійснювати пошук у ньому, виконувати розширений пошук за певними критеріями з можливістю експорту в Excel.

1. ЗАДАЧА ПОБУДОВИ АВТОМАТИЗОВАНОЇ СИСТЕМИ З ОБЛІКУ ДАНИХ АБІТУРІЄНТІВ

Метою даної роботи є створення системи, яка автоматизує роботу приймальної комісії.

Задачі, які потрібно розв’язати для досягнення мети:

1. Провести аналіз існуючих процедур роботи ПК.
2. Систематизувати структуру даних абітурієнтів.
3. Виділити завдання, які мають бути реалізовані в системі.
4. Обрати інструменти для реалізації.
5. Розробити систему.
6. Спроектувати інтерфейс для системи.
7. Протестувати роботу.

1.1. Дослідження запитів майбутніх користувачів системи

Майбутніми користувачами системи будуть співробітники ПК. Система планується як внутрішня для університету. Доступ надаватиметься тітьки співробітника. Не планується робити варіант для загального доступу для абітурієнтів або студентів. Дані мають бути доступні тільки для причетних людей. Інтерфейс слід спроектувати простою і наочною архітектурою [12]. Потрібна можливість створювати і обробляти особові справи. Планується використання на пристроях з різними ОС. Доступ має відбуватись швидко, без довгих додаткових налагоджувань.

1.2. Систематизація завдань, які мають виконуватись системою

Першочерговим завданням системи є збереження даних абітурієнтів. Доступ до інформації її подальша обробка – це обов'язкові пункти. Отож можна виділити такі задачі:

- Створення електронної особової справи абітурієнта
- Редагування існуючих справ
- Перегляд деталей справ
- Видалення справ
- Створення записів про результати вступних випробувань
- Обчислення рейтингу абітурієнтів
- Пошук по загальному списку
- фільтрація даних по заданих параметрах
- виведення фільтрованих
- експорт в Microsoft Excel фільтрованих даних

Пошук слід проводити за прізвищем.

Має бути можливість фільтрувати за кількома параметрами одночасно. Фільтрувати слід за наступними параметрами:

- Факультет
- Спеціальність
- Група
- Рівень освіти
- Кошти
- Чи потрібен гуртожиток
- Чи є іноземцем
- Чи має пільги
- Середній бал
- Місто проживання

1.3. Дослідження структури даних абітурієнтів

В результаті дослідження було виявлено, що необхідно зберігати наступну інформацію про абітурієнтів:

- ПІБ
- Факультет
- Спеціальність
- Група
- Рівень освіти(бакалавр/ магістр)
- Кошти (бюджет/контракт)
- Чи потрібен гуртожиток
- Чи є іноземцем
- Чи має пільги
- Дата народження
- Паспорт
- Ідентифікаційний код
- Адреса прописки
- Адреса проживання
- Email
- Домашній номер
- Мобільний номер
- Номер батьків
- Середній бал

З них обов’язковими для заповнення при створені особової справи є:

- ПІБ
- Рівень освіти(бакалавр/ магістр)
- Дата народження

- Паспорт
- Ідентифікаційний код
- Мобільний номер
- Номер батьків

Для зменшення можливих помилок при заповненні потрібно налаштувати перевірку даних при збережені. Очікується необхідність пошуку по населеному пункту, для адреси слід розробити структуру для збереження.

Також мають зберігатись результати вступних випробувань. Для кожного результату слід записувати ідентифікатор абітурієнта і інформацію про предмет. Середній бал буде розраховуватись на основі цих записів.

2. ЗАСОБИ РЕАЛІЗАЦІЇ ПРОГРАМНОЇ СИСТЕМИ

У наш час існує багато різних засобів і підходів для реалізації програмного продукту. Оскільки вибір дійсно широкий, виникає необхідність, перед розробкою, виділити час на підбір інструментів, оптимальних для конкретного проєкту.

2.1. Обґрунтування вибору веб-інтерфейсу

Для систем, у яких одночасно можуть працювати багато користувачів, є дві основні можливості реалізації користувацької частини – клієнтський додаток, що встановлюється на пристрій користувача, або веб-інтерфейс, доступ до якого здійснюється через браузер.

Для використання додатку необхідна інсталяція. Підчас розробки доводиться створювати окремі версії сумісні з різним ОС. Також користувач матиме доступ до системи тільки з пристроїв, де додаток установлений. Ці проблеми можна уникнути, якщо вибрати веб-додаток. Для його використання, користувачу достатньо мати веб-браузер на пристрої і знати адресу веб-сайту. У всіх сучасних ОС браузер інсталюється за замовчуванням.

Передбачається, що користувач працюватиме з актуальними даними, тому для роботи потрібне підключення до серверу. Це може бути як локальна, так і глобальна мережа. Дані абітурієнтів зберігатимуться і оброблятимуться на серверній частині системи, це означає кращий захист і малі вимоги до користувацьких пристроїв.

2.2. Обґрунтування вибору середовища розробки

Visual Studio [4] – це популярне середовище, розроблене Microsoft. VS включає підтримку налагодження, виділення синтаксису, інтелектуальне завершення коду, фрагменти та рефакторинг коду. Воно гнучко налаштовується, що дозволяє користувачам змінювати тему, комбінації клавіш, налаштування та встановлювати розширення, що додають функціональність.

Середовище VS включає не тільки редактор коду, а й багато інших корисних можливостей. Серед них є зручний пошук і управління додатковими бібліотеками, які значно спрощують роботу.

В наш час над програмним продуктом зазвичай працює не одна людина, а ціла команда. Для цього у VS є кілька інструментів: вбудований Git, що став незамінною частиною роботи над спільними проектами; система управління версіями Team Foundation (TFVC), а також Live Share, що дає можливість займатись спільною розробкою в режимі реального часу.

Роботу над інтерфейсами зручно виконувати у вбудованому дизайнері.

У VS можна працювати з базами даних. Це дуже зручно, що не доводиться використовувати додаткові менеджери для роботи з БД.

Усі частини проекту (код, інтерфейс, БД) доступні в одному середовищі.

2.3. Обґрунтування вибору платформи ASP.NET Core

ASP.NET Core є кросплатформеним, високопродуктивним середовищем з відкритим вихідним кодом для створення сучасних хмарних додатків, підключених до Інтернету. Дозволяє виконувати наступні завдання[1]:

- Створювати веб-додатки та служби, додатки Інтернету речей (IoT) і серверні частини для мобільних додатків.
- Використовувати засоби розробки в Windows, macOS і Linux.

- Виконувати розгортання в хмарі або локальному середовищі.

ASP.NET Core стабільно потрапляє в списки найбільш популярних, перспективних і використовуваних.

Надає наступні переваги:

- Єдине рішення для створення призначеного для користувача веб-інтерфейсу і веб-API
- Розроблено для тестованості
- Razor Pages спрощує написання коду для сценаріїв сторінок і підвищує його ефективність
- Blazor дозволяє використовувати в браузері мову C# разом з JavaScript
- Спільне використання серверної і клієнтської логік додатків, написаних за допомогою .NET
- Можливість розробки і запуску в ОС Windows, macOS і Linux
- Відкритий вихідний код і орієнтація на співтовариство
- Інтеграція сучасних клієнтських платформ і робочих процесів розробки
- Підтримка розміщення служб віддаленого виклику процедур (RPC) за допомогою gRPC
- Хмарна система конфігурації на основі середовища
- Вбудоване введення залежностей
- Спрощений високопродуктивний модульний конвеєр HTTP-запитів
- Управління паралельними версіями
- Інструментарій, що спрощує процес сучасної веб-розробки

ASP.NET Core MVC [1] являє собою спрощену, зручну для тестування платформу, оптимізовану для використання з ASP.NET Core.

ASP.NET Core MVC використовується при створенні додатків з шаблоном MVC (Рисунок 2.3) (Model View Controller - модель-вигляд-контролер).

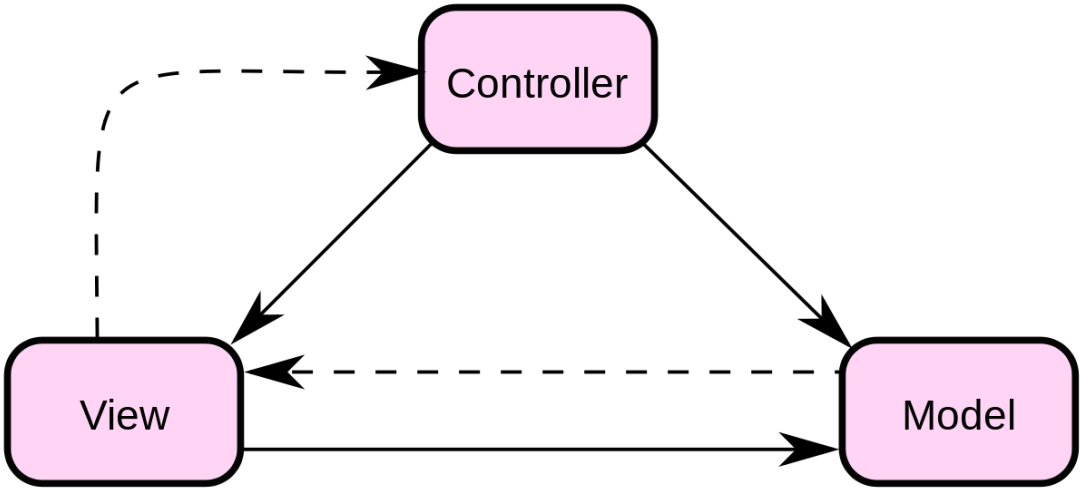


Рисунок 2.3 – Загальна схема шаблону MVC

Зручність тестування забезпечується поділом структури проекту на три основні частини: модель, вигляд, контролер. У частині моделей записується бізнес-логіка. Вигляд відповідає за подання вмісту через призначений для користувача інтерфейс. Використовується Razor обробник вигляду для впровадження коду .NET в розмітку HTML. Вигляд повинен мати мінімальну логіку. Контролери - це компоненти для управління взаємодією з користувачем, роботи з моделлю і вибору вигляду для відображення. У додатку MVC вигляд служить тільки для відображення інформації. Обробку введених даних, формування відповіді і взаємодія з користувачем забезпечує контролер.

ASP.NET Core Identity [7] :

- Це API, що підтримує функцію входу в користувальницький інтерфейс.
- Управляє користувачами, паролями, даними профілювання, ролями, твердженнями, маркерами, підтвердженням електронної пошти і т. д.

Identity зазвичай налаштовується за допомогою SQL Server бази даних для зберігання імен користувачів, паролів і даних профілів.

Наявність цих розширень (MVC, Identity) значно спрощує розробку програмного продукту. Також компанія Microsoft створила документацію, з якою дуже зручно вивчати і використовувати зазначені платформи.

2.4. Обґрунтування вибору системи керування базою даних

Для запланованих даних кращим способом зберігання є використання реляційної БД. Компанія Microsoft – це гігант в IT-сфері. Вона створила цілу екосистему для розробників програмних продуктів. MSSql частина цієї екосистеми і оскільки у вибраних для розробки платформ і цієї системи управління БД один розробник, їх взаємодія добре налаштована. Також у VS має вбудовані інструменти для роботи з MSSql, що є значною перевагою [2].

Плюси MSSql:

- СУБД масштабується, тому працювати з нею можна на портативних ПК або потужної мультипроцесорної техніці. Процесор може одночасно обробляти великий обсяг запитів.
- Розмір сторінок - до 8 кб, тому дані витягуються швидко, детальну і складну інформацію зберігати зручніше. Система дозволяє обробляти транзакції в інтерактивному режимі, є динамічне блокування.
- Рутинні адміністративні завдання автоматизовані: це управління блокуваннями, пам'яттю, редагування розмірів файлів. У системи продумані налаштування, можна створити профілі користувачів.
- Реалізовано пошук по фразах, тексту, словами, можна створювати ключові індекси.

2.5. Обґрунтування вибору додаткових бібліотек

Використання додаткових бібліотек допомагає значно спростити розробку програмного продукту. Використання наступної бібліотеки дозволить абстрагуватись від роботи з БД на пряму, натомість дасть додаткову гнучкість у модифікації БД.

Entity Framework (EF) Core [3] - це проста, кросплатформена і придатна до розширення версія популярної технології доступу до даних Entity Framework з відкритим вихідним кодом.

EF Core може використовуватися в якості об'єктно-реляційного модуля зіставлення (O / RM), який:

- Дозволяє розробникам .NET працювати з базою даних за допомогою об'єктів .NET.
- Усуває необхідність в більшій частині коду для доступу до даних, який зазвичай припадає писати.

EF Core підтримує безліч систем баз даних.

У EF Core доступ до даних здійснюється за допомогою моделі. Модель складається з класів сутностей і об'єкта контексту, який представляє сеанс взаємодії з базою даних. Об'єкт контексту дозволяє виконувати запити і зберігати дані.

EF підтримує такі підходи до розробки моделей:

- Створення моделі на основі існуючої бази даних.
- Написання коду моделі вручну відповідно до бази даних.
- Міграції EF для створення бази даних на основі попередньо створеної моделі.

Міграції дозволяють розвивати базу даних у міру зміни моделі.

Ще одне готове рішення яке допоможе у розробці – це бібліотека EPPlus. EPPlus [6] – це бібліотека .NET Framework / .NET Core для управління електронними таблицями Office Open XML, поширювана через Nuget. Вона має безкоштовну ліцензію для некомерційного використання. Цю бібліотеку постійно оновлюють. Її функціонал дає можливість використовувати майже всі можливості MS Excel. З її допомогою можна формувати статистику і звіти в відповідному форматі.

3. ОПИС ПРОГРАНМОЇ РЕАЛІЗАЦІЇ

3.1. Архітектура системи

Вебзастосунок побудований на основі шаблону MVC (модель-вигляд-контролер). Загальний варіант взаємодії елементів продемонстровано на рисунку (Рисунок 4.1.1 – Загальна схема взаємодії елементів вебзастосунку з базою даних за шаблоном MVC).

Основна модель у системі – це особова справа абітурієнта. Вона водночас є прототипом для таблиці в базі даних. Основні функції системи направлені на додавання/ редагування/ пошук записів у цій таблиці.

Також у системи є два основні Вигляди, з якими користувач взаємодіятиме найчастіше. Перший – це відображення усіх записів про абітурієнтів у табличному вигляді і можливість пошуку/ фільтрування у цій таблиці за певними критеріями. Другий – це додавання/редагування особових справ.

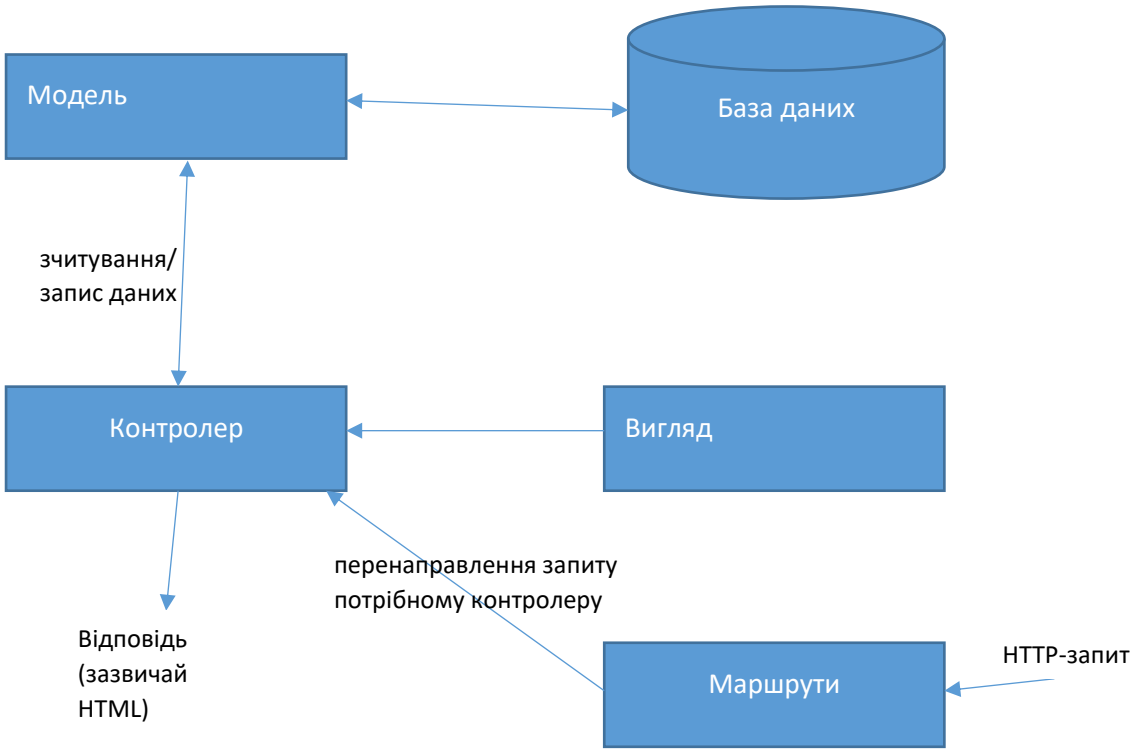


Рисунок 3.1 – Загальна схема взаємодії елементів вебзастосунку з базою даних за шаблоном MVC

За взаємодію між даними і виведені їх у Вигляді відповідає контролер. У розроблюваної системи один основний контролер. У ньому прописані функції взаємодії з базою даних, такі як: додавання/редагування записів в таблиці «особові справи», пошук та фільтрація записів.

3.2. Опис структури бази даних

Система обліку даних абітурієнтів звісно потребує базу даних. Для цього вебзастосунку вибрана база даних на MSSql. Для спрощення роботи з БД використовується Entity Framework (EF). EF дає можливість працювати з БД через класи(моделі). При створені БД використовувався підхід “спочатку код”, тобто спочатку були створені класи(моделі), а потім з допомогою міграції ініціалізувалась БД на основі створених перед тим моделей. Міграції у EF також дають можливість модифікувати структуру БД без втрати даних, що стане у нагоді при розширені системи.

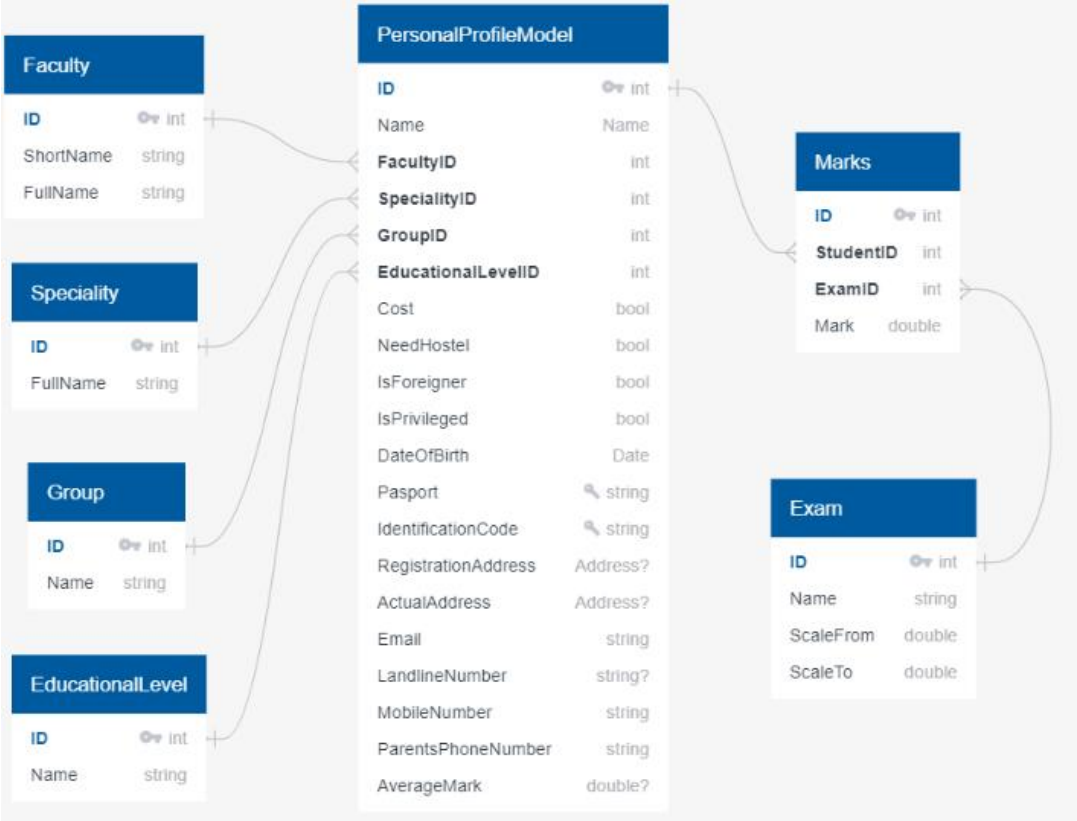


Рисунок 3.2 – Структура бази даних системи обліку даних абітурієнтів

Структура бази даних системи обліку даних абітурієнтів наведена на рисунку 3.2. БД складається з двох основних таблиць: Особова справа абітурієнта (PersonalProfileModel) і Оцінки з вступних іспитів (Marks); і п’яти довідкових таблиць: Факультет, Спеціальність, Група, Рівень освіти (EducationalLevel) та Вступні іспити(Exam).

3.3. Опис процесів системи

По шаблону MVC було створено моделі для об’єктів системи. З допомогою Entity Framework установлюється з’єднання з сервером і підключається БД, з’єднання є асинхроним. Для веб- додатку були згенерований контролер і вигляд, за підійшовшим шаблоном. Після початкового з’єднання за усі процеси в системі

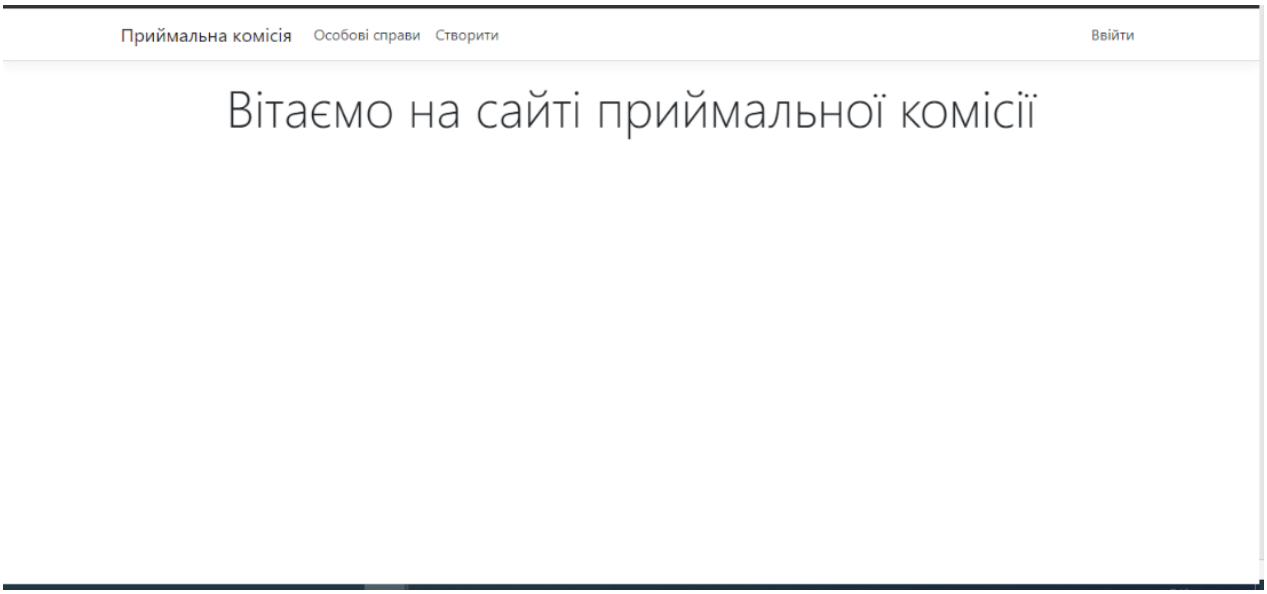
відповідає головний контролер. Він підвантажує дані для користувацького інтерфейсу, створює запити до БД. Перемальовує інтерфейс при виконанні різних функцій. У головному контролері відбувається формування Excel файлу.

4. ОПИС ПРОГРАМНОЇ РЕАЛІЗАЦІЇ

4.1. Опис роботи з веб-інтерфейсом

Для початку роботи з веб-додатком необхідно в браузері зайти на відповідну сторінку.

Домашня сторінка виглядає як показано на Рисунку 4.1.1

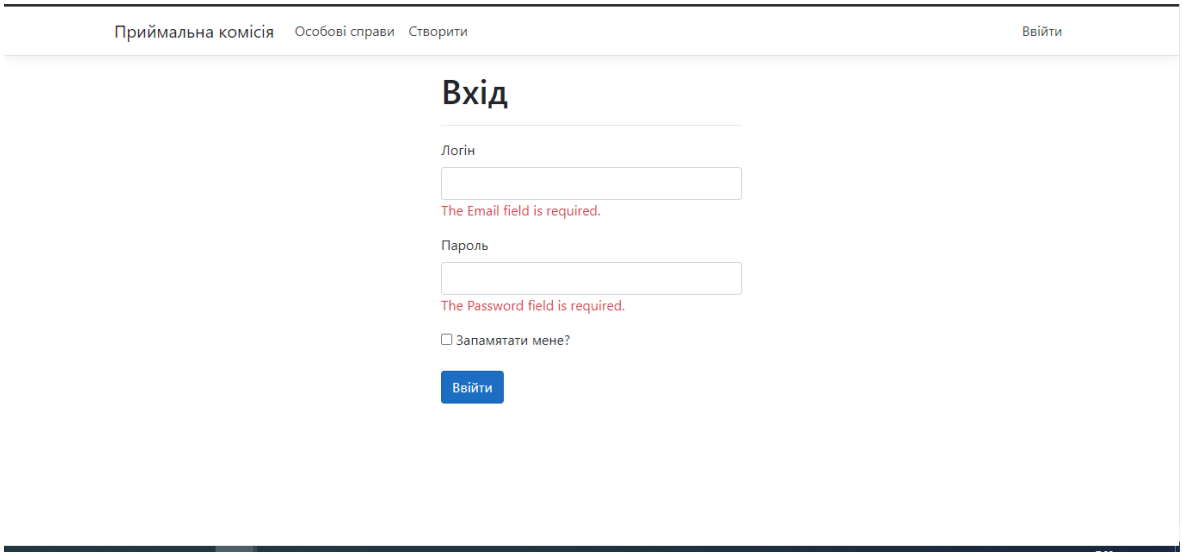


Рисунку 4.1.1 – Скриншот домашньої сторінки

Зверху відображаються посилання на веб-інтерфейси «Особові справи» – перелік справ з інструментами обробку, «Створити» - інтерфейс створення справи абітурієнта.

Для того щоб створити справу необхідно пройти авторизацію. Для цього можна перейти за посиланням «Вхід», що знаходиться у правому верхньому куті.

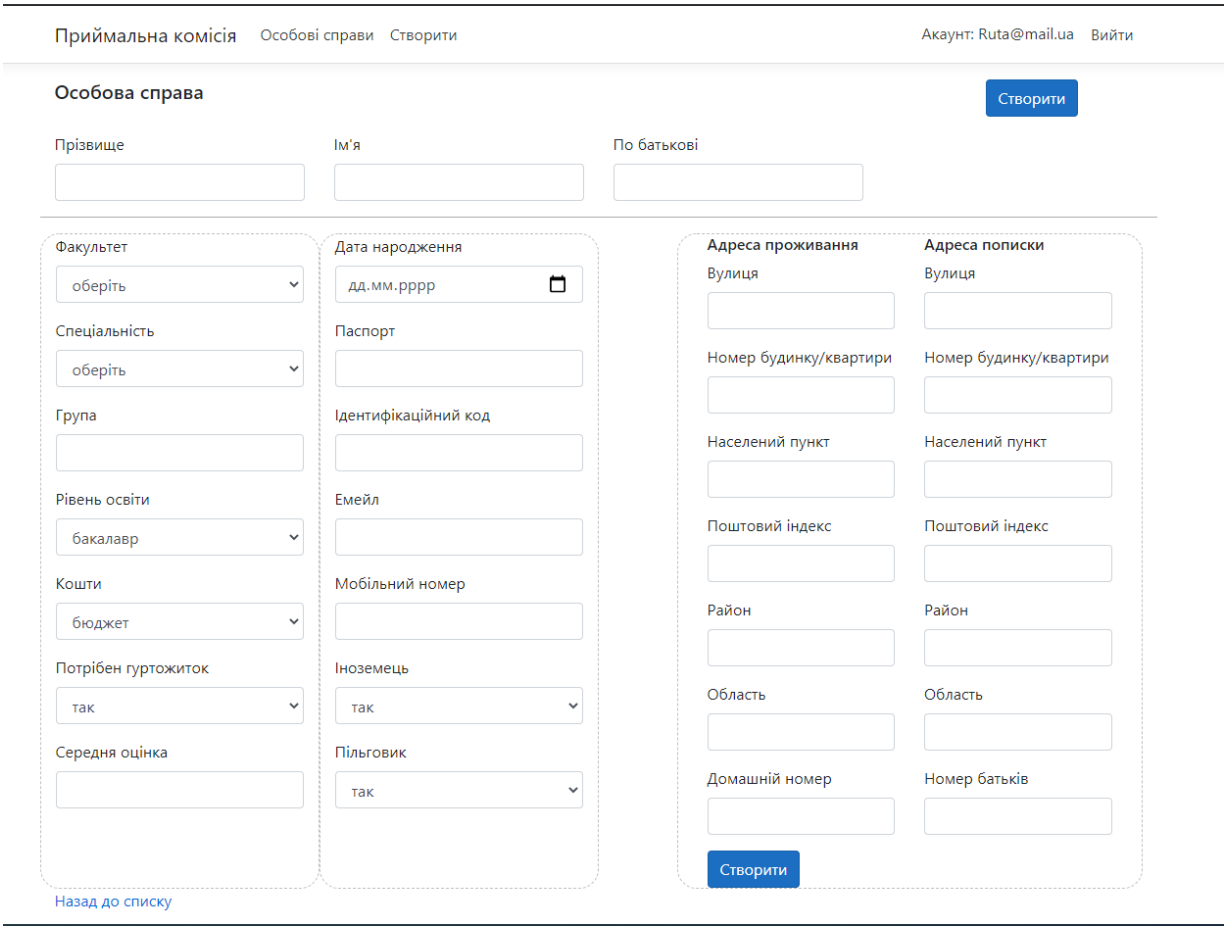
За посиланням знаходиться сторінка вигляду як на Рисунку 4.1.2.



Рисунку 4.1.2 – Скриншот інтерфейсу входу

У випадку якщо неавторизований користувач спробує перейти на інтерфейс створення, його перешле на сторінку авторизації.

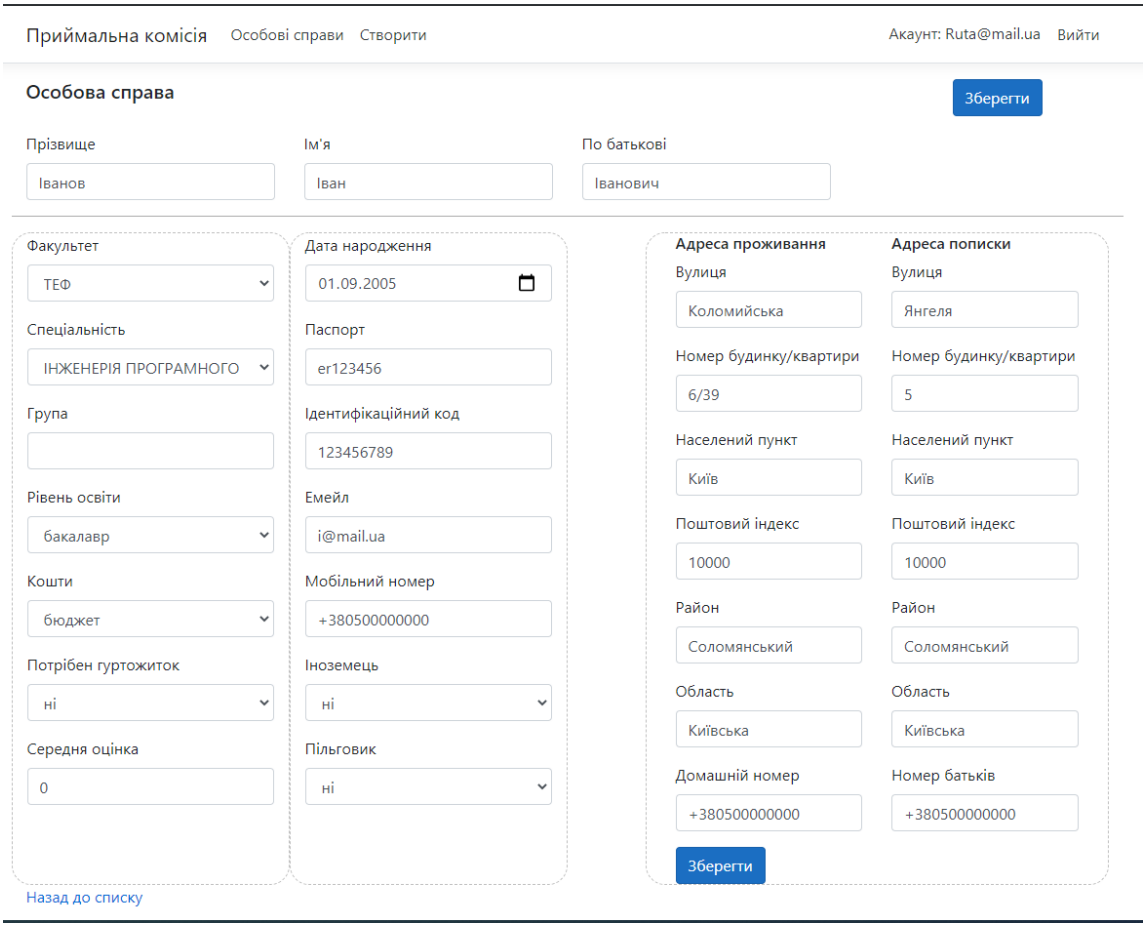
Далі на Рисунку 4.1.3 представлений вигляд інтерфейсу створення особової справи.



Рисунку 4.1.3 – Скриншот інтерфейсу створення особової справи абітурієнта

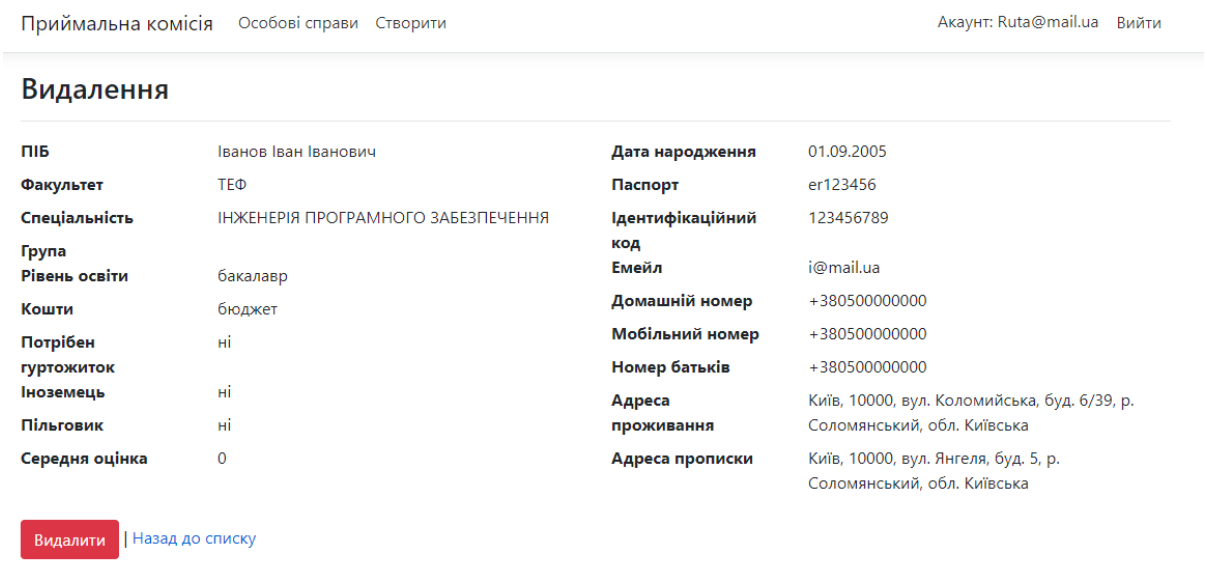
При спробі створення справи з незаповненими обов’язковими полями з’являтиметься попередження, створення можливе тільки тоді, коли введені дані відповідають очікуваному формату.

На Рисунку 4.1.4 продемонстровано інтерфейс редагування особових справ. дані підтягуються автоматично.



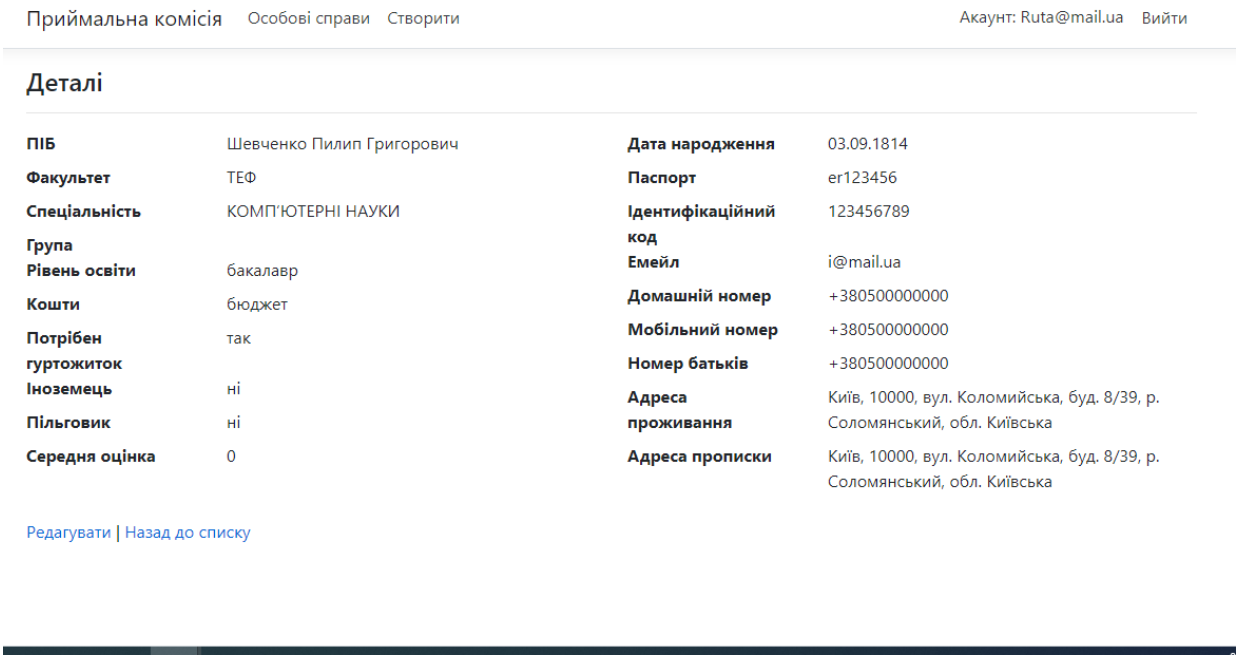
Рисунку 4.1.4 – Скриншот інтерфейсу редагування особової справи абітурієнта

Перед видаленням показуються деталі обраної справи (Рисунку 4.1.5).



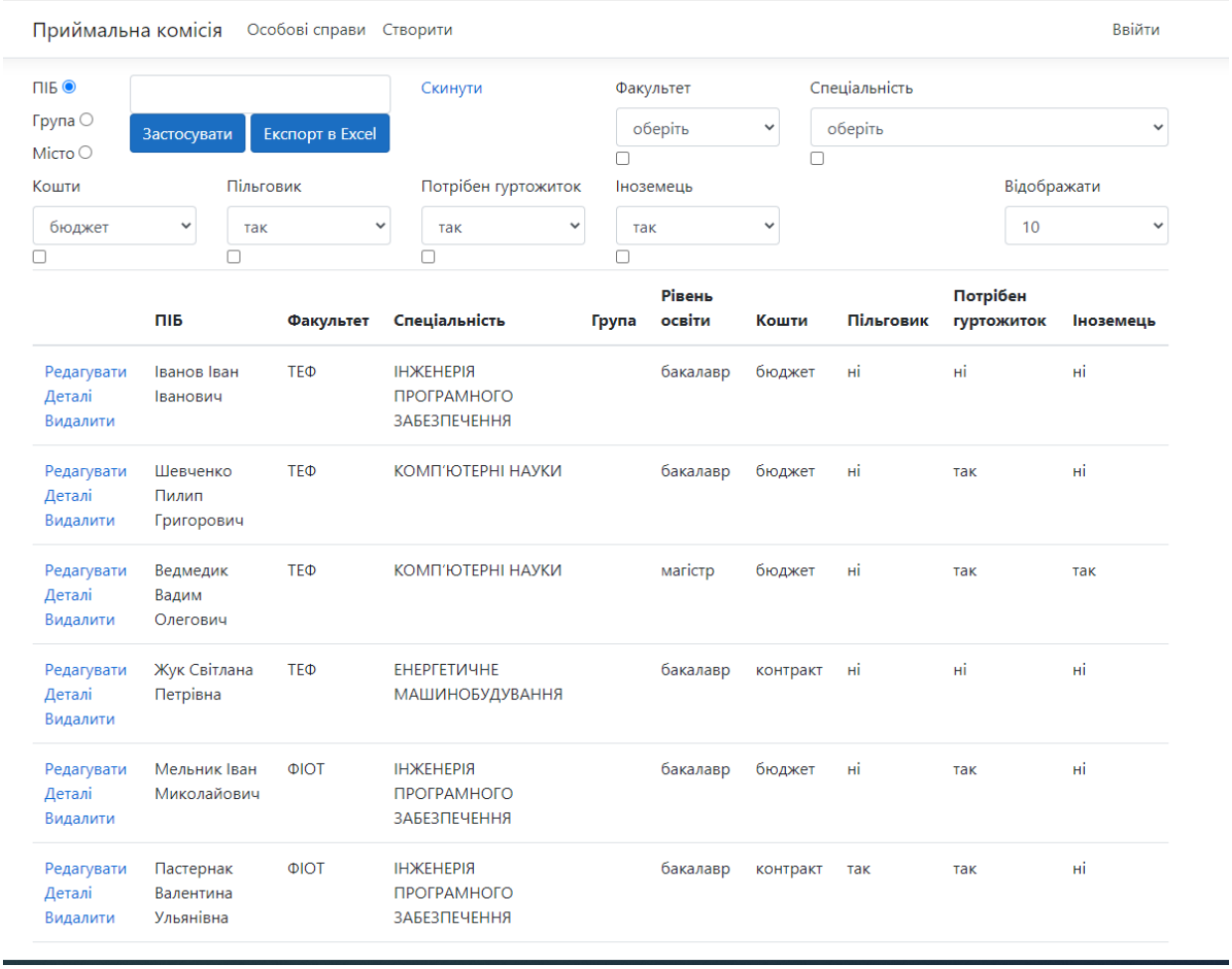
Рисунку 4.1.5 – Скриншот інтерфейсу видалення особової справи абітурієнта

Далі продемонстровано інтерфейсу перегляду деталей справи абітурієнта



Рисунку 4.1.5 – Скриншот інтерфейсу видалення особової справи абітурієнта

Основним інтерфейсом для роботи з системою є «Особові Справи»(Рисунку 4.1.6).



Рисунку 4.1.6 – Скриншот інтерфейсу «Особові Справи»

В шапці знаходяться інструменти керування пошуком, фільтруанням і експортом даних. Цей інтерфейс доступний для неавторизованих користувачів, тут міститься тільки базова інформація про абітурієнтів. З нього можна перейти до редагування, видалення, перегляду деталей справ. Функції редагування, видалення, перегляду деталей, експорту фільтрованих даних доступні тільки авторизованим користувачам.

4.2. Опис роботи адміністратора системи

Оскільки у веб-додатку не передбачено механізму реєстрації нових користувачів, створювати облікові записи, а також допомагати з відновленням даних

облікових записів має адміністратор системи. У проекті є інструменти для створення потрібних для цього інтерфейсів. Отже при необхідності їх можна реалізувати. А до реалізації адміністратор має напряму взаємодіяти з базою даних користувачів, з допомогою програмного середовища, яке вміє працювати з БД на MSSql.

У випадку якщо потрібно буде модифікувати структуру БД адміністратор може створити міграцію з допомогою Entity Framework.

Додати новий функціонал до веб-додатку не складно. Достатньо створити нові Model, View і Controller, а потім розмістити посилання і модифікувати мапу додатку.

ВИСНОВКИ

Результатом дипломної роботи є створення автоматизованої системи з обліку даних абітурієнтів. Було спроектовано і реалізовано веб-додаток, який автоматизує роботу приймальної комісії.

В результаті аналізу предметної області були сформульовані основні вимоги до системи. На основі отриманих знань була спроектована структура реляційної бази даних, обрані технології для реалізації веб-додатку.

Система дає змогу користувачу створювати електронну особову справу абітурієнта, вести облік результатів вступних випробувань, здійснювати пошук, сортування та виведення даних по заданих параметрах, готувати дані для звітів і статистики з можливістю експорту в Microsoft Excel.

Оскільки зберігаються приватні дані абітурієнтів, для їх захисту була реалізована авторизація. Прорoблено розподілення доступу для авторизованих і не авторизованих користувачів.

Адміністратор веб-додатку має можливість додавати користувачів. При необхідності можливо виконати модифікацію БД.

Для системи був спроектований зрозумілий і легкий у користуванні інтерфейс.

Розробка виконувалась в середовищі розробки Visual Studio з використанням мови C# та фреймворку ASP.NET Core MVC. Використання Entity Framework спрощує подальше розширення БД. Шаблон MVC дає змогу без великих зусиль розширювати в майбутньому функціонал системи.

Для веб-додатку можуть бути актуальні наступні доробки:

- створення можливості реєстрації на сайті
- формування типових звітів
- створення сховища для електронних копій документів абітурієнтів.

Система має високі показники надійності, ефективності та орієнтації на споживача. Таким чином, досягнута мета, поставлена в даній дипломній роботі, а саме: створення автоматизованої системи з обліку даних абітурієнтів.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. ASP.NET Core MVC [Електронний ресурс] – Режим доступу: https://docs.microsoft.com/en-gb/aspnet/core/mvc/overview?view=aspnetcore-5.0
2. MSSql [Електронний ресурс] – Режим доступу: https://docs.microsoft.com/en-us/sql/?view=sql-server-ver15
3. Entity Framework [Електронний ресурс] – Режим доступу: https://docs.microsoft.com/en-gb/ef/
4. Visual Studio [Електронний ресурс] – Режим доступу: https://docs.microsoft.com/en-gb/visualstudio/?view=vs-2019
5. HTML [Електронний ресурс] – Режим доступу: https://devdocs.io/html/
6. EPPlus [Електронний ресурс] – Режим доступу: https://www.epplussoftware.com/docs/5.6/index.html
7. ASP.NET Core Identity [Електронний ресурс] – Режим доступу: https://docs.microsoft.com/en-gb/aspnet/core/security/authentication/?view=aspnetcore-5.0
8. Razor [Електронний ресурс] – Режим доступу: https://docs.microsoft.com/en-gb/aspnet/core/mvc/views/razor?view=aspnetcore-5.
9. Кайт, Томас. Oracle для професионалов. Архитектура, методики программирования и основные особенности версий 9i и 10g / Томас Кайт ; [пер. с англ. Я. П. Волковой, Ю. И. Корниенко, Н. А. Мухина]. - Москва [и др.] : Вильямс, 2007. - 839 с. : табл.
10. Кормен, Томас Г., автор. Вступ до алгоритмів : переклад з англійської третього видання / Томас Г. Кормен, Чарльз Е. Лейзерсон, Роналд Л. Рівест, Кліфорд Стайн. - Київ : К.І.С., 2019. - 1286 сторінок : рисунки, таблиці.
11. Скит, Джон. C# : программирование для профессионалов / Джон Скит ; [предисловие Эрика Липперта]. - Москва [и др.] : Вильямс, 2011. - 544 с. : ил., табл.

12. Купер, Алан Психбольница в руках пациентов / Алан Купер; [пер. с англ. Михайло Зісліса]. - 2011. – 336с. : ил.

ДОДАТОК А

Автоматизована система з обліку даних абітурієнтів

Специфікація

УКР.НТУУ”КПІ”_ТЕФ_АПЕПС_ТР62146_21Б

Аркушів 2

Київ — 2021

Позначення	Найменування	Примітки
Документація		
УКР.НТУУ”КПІ”_ТЕФ_АПЕПС_ ТР62146_21Б	Записка.docx	Текстова частина дипломної роботи
Компоненти		
УКР.НТУУ”КПІ”_ТЕФ_АПЕПС_ ТР62146_21Б 12-1	PersonalProfileModelsController.cs	Компонент, що містить функції обробки взаємодії користувача з інтерфейсом, алгоритми роботи з базою даних
УКР.НТУУ”КПІ”_ТЕФ_АПЕПС_ ТР62146_21Б 12-2	PersonalProfileModel.cs	Компонент, що містить структуру даних про абітурієнта

ДОДАТОК Б

Автоматизована система з обліку даних абітурієнтів

Текст програми

УКР.НТУУ”КПІ”_ТЕФ_АПЕПС_ ТР62146_21Б 12-1

Аркушів 13

Київ — 2021

PersonalProfileModelsController.cs

```
using System;
using System.Collections.Generic;
using System.IO;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Authorization;
using Microsoft.AspNetCore.Mvc;
using Microsoft.AspNetCore.Mvc.Rendering;
using Microsoft.EntityFrameworkCore;
using OfficeOpenXml;
using PK_KPI0.Data;
using PK_KPI0.Models;

namespace PK_KPI0.Controllers
{
    [Authorize]
    public class PersonalProfileModelsController : Controller
    {
        private readonly UniContext _context;

        public PersonalProfileModelsController(UniContext context)
        {
            _context = context;
        }

        // GET: PersonalProfileModels
        [AllowAnonymous]
        public async Task<IActionResult> Index(int pageSize, string searchString, string SearchIn, string currentFilter, string filter, int vfac, int vspec, int vcost, int vpriv, int vhost, int vfori, bool fac, bool spec, bool cost, bool priv, bool host, bool fori, int? pageNumber)
        {
            ViewData["CurrentFilter"] = searchString;

            if (searchString != null)
```

```
        {
            pageNumber = 1;
        }
        else
        {
            searchString = currentFilter;
        }

        var students = from s in _context.PersonalProfileModels
                       select s;
        if (!String.IsNullOrEmpty(searchString)&&SearchIn.Equals("Name"))
        {
            students = students.Where(s => s.Name.Surname.Contains(searchString)
                                || s.Name.FirstName.Contains(searchString) ||
s.Name.MiddleName.Contains(searchString));
        }

        if (!String.IsNullOrEmpty(searchString) && SearchIn.Equals("Group"))
        {
            students = students.Where(s => s.Group.Contains(searchString)
                                || s.Group.Contains(searchString) || s.Group.Contains(searchString));
        }
        if (!String.IsNullOrEmpty(searchString) && SearchIn.Equals("City"))
        {
            students = students.Where(s => s.ActualAddress.City.Contains(searchString));
        }
        if (fac)
        {
            students = students.Where(s => (int)s.Faculty==vfac);
        }
        if (spec)
        {
            students = students.Where(s => (int)s.Speciality == vspec);
        }
        if (cost)
```

```
{
   students = students.Where(s => (int)s.Cost == vcost);
}
if (priv)
{
   students = students.Where(s => (int)s.IsPrivileged == vpriv);
}
if (host)
{
   students = students.Where(s => (int)s.NeedHostel == vhost);
}
if (fori)
{
   students = students.Where(s => (int)s.IsForeigner == vfori);
}

if (filter!=null && filter.Equals("Експорт в Excel"))
{
   ExcelPackage.LicenseContext = LicenseContext.NonCommercial;
   using (ExcelPackage excelPackage = new ExcelPackage())
   {
      //Set some properties of the Excel document
      excelPackage.Workbook.Properties.Author = "VDWWD";
      excelPackage.Workbook.Properties.Title = "Перелік студентів";
      excelPackage.Workbook.Properties.Created = DateTime.Now;

      //Create the WorkSheet
      ExcelWorksheet worksheet = excelPackage.Workbook.Worksheets.Add("Сторінка 1");

      //заголовок таблиці
      worksheet.Cells[1,1].Value = "ID";
      worksheet.Cells[1,2].Value = "ПІБ";
      worksheet.Cells[1,3].Value = "Факультет";
      worksheet.Cells[1,4].Value = "Спеціальність";
      worksheet.Cells[1,5].Value = "Група";
```

```
            worksheet.Cells[1,6].Value = "Рівень освіти";
            worksheet.Cells[1,7].Value = "Кошти";
            worksheet.Cells[1,8].Value = "Потрібен гуртожиток";
            worksheet.Cells[1,9].Value = "Іноземець";
            worksheet.Cells[1,10].Value = "Пільговик";
            worksheet.Cells[1,11].Value = "Дата народження";
            worksheet.Cells[1,12].Value = "Паспорт";
            worksheet.Cells[1,13].Value = "Ідентифікаційний код";
            worksheet.Cells[1,14].Value = "Адреса прописки";
            worksheet.Cells[1,15].Value = "Адреса проживання";
            worksheet.Cells[1,16].Value = "Емейл";
            worksheet.Cells[1,17].Value = "Домашній номер";
            worksheet.Cells[1,18].Value = "Мобільний номер";
            worksheet.Cells[1,19].Value = "Номер батьків";
            worksheet.Cells[1,20].Value = "Середній бал";

            //заповнення excel
            int i = 2;
            foreach(PersonalProfileModel student in students)
            {
               worksheet.Cells[i,1].Value = student.ID;
               worksheet.Cells[i, 2].Value = student.Name.Surname + " " + student.Name.FirstName + "
" + student.Name.MiddleName;
               worksheet.Cells[i, 3].Value = student.Faculty;
               worksheet.Cells[i, 4].Value = student.Speciality;
               worksheet.Cells[i, 5].Value = student.Group;
               worksheet.Cells[i, 6].Value = student.EducationalLevel;
               worksheet.Cells[i, 7].Value = student.Cost;
               worksheet.Cells[i, 8].Value = student.NeedHostel;
               worksheet.Cells[i, 9].Value = student.IsForeigner;
               worksheet.Cells[i, 10].Value = student.IsPrivileged;
               worksheet.Cells[i, 11].Value = student.DateOfBirth.ToString("dd.MM.yyyy");
               worksheet.Cells[i, 12].Value = student.Pasport;
               worksheet.Cells[i, 13].Value = student.IdentificationCode;
               try
```

```
                {
                    worksheet.Cells[i, 14].Value = student.ActualAddress.City + " " +
student.ActualAddress.ZIPCode + ", вул. " + student.ActualAddress.Street + " буд." +
student.ActualAddress.HouseNumFlatNum + ", р." + student.ActualAddress.District + ", обл. " +
student.ActualAddress.Region;
                }
                catch { }
                try
                {
                    worksheet.Cells[i, 15].Value = student.RegistrationAddress.City + " " +
student.RegistrationAddress.ZIPCode + ", вул. " + student.RegistrationAddress.Street + " буд." +
student.RegistrationAddress.HouseNumFlatNum + ", р." + student.RegistrationAddress.District + ", обл.
" + student.RegistrationAddress.Region;
                }
                catch { }
                worksheet.Cells[i, 16].Value = student.Email;
                worksheet.Cells[i, 17].Value = student.LandlineNumber;
                worksheet.Cells[i, 18].Value = student.MobileNumber;
                worksheet.Cells[i, 19].Value = student.ParentsPhoneNumber;
                worksheet.Cells[i, 20].Value = student.AverageMark;

                i++;
            }

            //Save your file
            FileInfo fi = new FileInfo(@"d:\File.xlsx");
            excelPackage.SaveAs(fi);
        }
        byte[] fileBytes = GetFile(@"d:\File.xlsx");
        return File(fileBytes, System.Net.Mime.MediaTypeNames.Application.Octet, @"d:\File.xlsx");

    }

    if(pageSize == 0) pageSize=10;
```

```
        return View(await PaginatedList<PersonalProfileModel>.CreateAsync(students.AsNoTracking(),
pageNumber ?? 1, pageSize));
    }

    byte[] GetFile(string s)
    {
        System.IO.FileStream fs = System.IO.File.OpenRead(s);
        byte[] data = new byte[fs.Length];
        int br = fs.Read(data, 0, data.Length);
        if (br != fs.Length)
            throw new System.IO.IOException(s);
        return data;
    }

    // GET: PersonalProfileModels/Details/5
    public async Task<IActionResult> Details(int? id)
    {
        if (id == null)
        {
            return NotFound();
        }

        var personalProfileModel = await _context.PersonalProfileModels
            .FirstOrDefaultAsync(m => m.ID == id);
        if (personalProfileModel == null)
        {
            return NotFound();
        }

        return View(personalProfileModel);
    }

    // GET: PersonalProfileModels/Create
    public IActionResult Create()
    {
```

```
        return View();
    }

    // POST: PersonalProfileModels/Create
    [HttpPost]
    [ValidateAntiForgeryToken]
    public async Task<IActionResult>
Create([Bind("Name,ID,Faculty,Speciality,EducationalLevel,NeedHostel,IsForeigner,IsPrivileged,DateO
fBirth,Pasport,IdentificationCode,RegistrationAddress,ActualAddress,Email,LandlineNumber,MobileNu
mber,ParentsPhoneNumber,AverageMark,Cost")] PersonalProfileModel personalProfileModel)
    {
        if (ModelState.IsValid)
        {
            _context.Add(personalProfileModel);
            await _context.SaveChangesAsync();
            return RedirectToAction(nameof(Index));
        }
        return View(personalProfileModel);
    }

    // GET: PersonalProfileModels/Edit/5
    public async Task<IActionResult> Edit(int? id)
    {
        if (id == null)
        {
            return NotFound();
        }

        var personalProfileModel = await _context.PersonalProfileModels.FindAsync(id);
        if (personalProfileModel == null)
        {
            return NotFound();
        }
        return View(personalProfileModel);
    }
```

```
// POST: PersonalProfileModels/Edit/5

[HttpPost]
[ValidateAntiForgeryToken]
public async Task<IActionResult> Edit(int id,
[Bind("Name,ID,Faculty,Speciality,EducationalLevel,NeedHostel,IsForeigner,IsPrivileged,DateOfBirth,Pasport,IdentificationCode,RegistrationAddress,ActualAddress,Email,LandlineNumber,MobileNumber,ParentsPhoneNumber,AverageMark,Cost")] PersonalProfileModel personalProfileModel)
{
   if (id != personalProfileModel.ID)
   {
      return NotFound();
   }

   if (ModelState.IsValid)
   {
      try
      {
         _context.Update(personalProfileModel);
         await _context.SaveChangesAsync();
      }
      catch (DbUpdateConcurrencyException)
      {
         if (!PersonalProfileModelExists(personalProfileModel.ID))
         {
            return NotFound();
         }
         else {  throw;   }
      }
      return RedirectToAction(nameof(Index));
   }
   return View(personalProfileModel);
}
```

```
        // GET: PersonalProfileModels/Delete/5
        public async Task<IActionResult> Delete(int? id)
        {
            if (id == null)           {  return NotFound();}

            var personalProfileModel = await _context.PersonalProfileModels
                .FirstOrDefaultAsync(m => m.ID == id);
            if (personalProfileModel == null)          {   return NotFound(); }

            return View(personalProfileModel);
        }

        // POST: PersonalProfileModels/Delete/5
        [HttpPost, ActionName("Delete")]
        [ValidateAntiForgeryToken]
        public async Task<IActionResult> DeleteConfirmed(int id)
        {
            var personalProfileModel = await _context.PersonalProfileModels.FindAsync(id);
            _context.PersonalProfileModels.Remove(personalProfileModel);
            await _context.SaveChangesAsync();
            return RedirectToAction(nameof(Index));
        }

        private bool PersonalProfileModelExists(int id)
        {
            return _context.PersonalProfileModels.Any(e => e.ID == id);
        }
    }
}
```

PersonalProfileModel.cs

```
using Microsoft.EntityFrameworkCore;
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.ComponentModel.DataAnnotations;
using System.Linq;
using System.Threading.Tasks;

namespace PK_KPI0.Models
{
    public class PersonalProfileModel
    {
        public int ID { get; set; }
        [Required]
        public Name Name { get; set; }

        [Display(Name = "Факультет")]
        public Faculties Faculty { get; set; } //факультет
        [Display(Name = "Спеціальність")]
        public Specialities Speciality { get; set; }
        [Display(Name = "Група")]
        public string Group { get; set; }
        [Required]
        [Display(Name = "Рівень освіти")]
        public EducationalLevel EducationalLevel { get; set; }
        [Display(Name = "Кошти")]
        public Costs Cost { get; set; }
        [Display(Name = "Потрібен гуртожиток")]
        public YesNo NeedHostel { get; set; }
        [Display(Name = "Іноземець")]
        public YesNo IsForeigner { get; set; }
        [Display(Name = "Пільговик")]
        public YesNo IsPrivileged { get; set; }
```

```
    [Required(ErrorMessage = "Введіть дату")]
    [DataType(DataType.Date)]
    [Display(Name = "Дата народження")]
    public DateTime DateOfBirth { get; set; }

    [Required(ErrorMessage = "Введіть номер паспорта")]
    [Display(Name = "Паспорт")]
    public string Pasport { get; set; }
    [Required(ErrorMessage = "Введіть ідентифікаційний код")]
    [Display(Name = "Ідентифікаційний код")]
    public string IdentificationCode { get; set; }
    [Display(Name = "Адреса прописки")]
    public Address RegistrationAddress { get; set; }
    [Display(Name = "Адреса проживання")]
    public Address ActualAddress { get; set; }
    [Display(Name = "Емейл")]
    public string Email { get; set; }

    [Display(Name = "Домашній номер")]
    public string LandlineNumber { get; set; }
    [Display(Name = "Мобільний номер")]
    [Required(ErrorMessage = "Введіть номер телефону")]
    public string MobileNumber { get; set; }
    [Required(ErrorMessage = "Введіть номер телефону")]
    [Display(Name = "Номер батьків")]
    public string ParentsPhoneNumber { get; set; }

    [Display(Name = "Середня оцінка")]
    public double? AverageMark { get; set; } //середній бал

    public ICollection<MarkModel> Marks { get; set; }
}

[Owned]
public class Address
```

```
    {
        public string Street { get; set; }
        public string HouseNumFlatNum { get; set; }
        public string City { get; set; }
        public string District { get; set; }
        public string Region { get; set; }
        public string ZIPCode { get; set; }
    }
    [Owned]
    public class Name
    {
        [Required(ErrorMessage = "Введіть прізвище")]
        public string Surname { get; set; }
        [Required(ErrorMessage = "Введіть ім'я")]
        public string FirstName { get; set; }
        [Required(ErrorMessage = "Введіть по батькові")]
        public string MiddleName { get; set; }
    }
}
```